### F:\ACADEMICS\MITE\MITE2015-16\Project co-ordinator\2016-1017\MITE Logo(Original JPEG).jpg

### MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

*Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified Institution*

*(A Unit of Rajalaxmi Education Trust®, Mangalore - 575001)*

Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi.

Badaga Mijar, Moodabidri-574225, Karnataka

**Project Report**

**On**

# “Traveling Assistant”

### Submitted By

|  |  |
| --- | --- |
| **Spoorthi G** | 4MT21AI055 |
| Aishwarya R Kulkarni | 4MT22AI401 |
| Renuprasad M | 4MT22AI405 |
| Karthik K R | 4MT21AI025 |
| Angith | 4MT21AI007 |

**DEPARTMENT OF**

**ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**

**2022-2023**

# ABSTRACT

By providing users thorough travel-related information and services, The Travelling Assistant is a cutting-edge computer program designed to enhance the travel experience. This abstract offers an overview of the key features and functions of the Travelling Assistant, an excellent travel companion.

The Travelling Assistant provides comprehensive details on a variety of travel destinations throughout the world. Users get access to each location's descriptions, images, historical data, forecasts for the weather, and popular attractions, enabling them in choosing accurate decisions about their trips

Users can use the Travelling Assistant to swiftly organize their trips. The system will generate personalized trip plans using suggested routes, places to stay, and activities as they input their travel choices, including budget, duration, and interests.

The Travelling Assistant allows customers book reservations on stuff including airfare, lodging, rentals automobiles, and activities. It offers an efficient booking procedure to ensure tourists can quickly book their accommodations.

**INTRODUCTION**

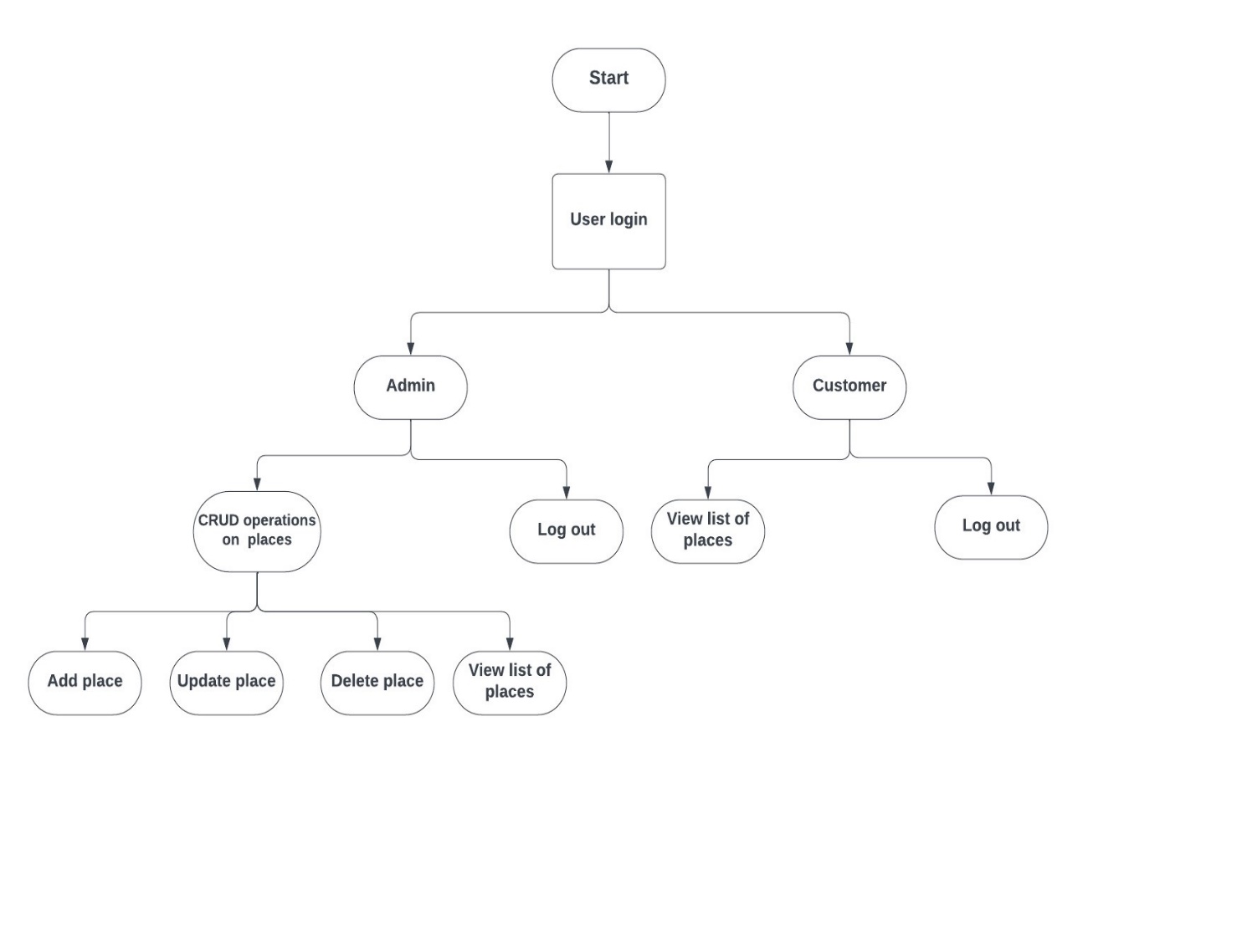
A software program named the Travelling Assistant Booking System enables customers arrange bookings for travel-related services including lodging, transportation, and tours. This approach seeks to offer comprehensive details on every location and services available in a particular region. The admin area and the Customer/User section make up its two main sections.

The whole database of locations and services in the targeted area is handled by the admin section of the application. To ensure the data's accuracy and relevancy, admin users can perform CRUD (Create, Read, Update, Delete) tasks on it. Adding new locations, updating existing data, eliminating out-of-date listings, and preserving user access are some of the key administrative functions.

Users receive access to and interactions with the travel-related information and services via the Customer/User section of the application. Customers can look up details regarding multiple locations, like descriptions, expenditures, and user reviews. Users get a chance to make reservations, investigate their previous reservations, and get notifications about their bookings. Only those who have registered can access the booking features according to the login system.

The Travelling Assistant Booking System in C functions as a central hub for customers to interact with travel-related services and information in the area in question. Customers can effortlessly explore and make reservations, while admins are able to supervise the contents. These features and functionalities are carried out by the program using C programming, giving travelers and travel service providers a user-friendly and efficient solution.

**FLOW CHART**



# MIND MAP

# 

**SOURCE CODE**

#include <stdio.h>

#include <string.h>

#define MAX\_PLACES 100

#define MAX\_BOOKINGS 100

// Structure for a place

struct Place {

char name[100];

};

// Structure for a booking

struct Booking {

char place\_name[100];

char customer\_name[100];

char booking\_date[20];

char booking\_time[20];

};

struct Place places[MAX\_PLACES];

struct Booking bookings[MAX\_BOOKINGS];

int num\_places = 0;

int num\_bookings = 0;

// Function to check admin login

int admin\_login() {

char username[100], password[100];

printf("Admin Login\n");

printf("Username: ");

scanf("%s", username);

printf("Password: ");

scanf("%s", password);

return strcmp(username, "admin") == 0 && strcmp(password, "adminpass") == 0;

}

// Function to save places and bookings to files

void save\_data() {

FILE \*places\_file = fopen("places.txt", "w");

if (places\_file) {

for (int i = 0; i < num\_places; i++) {

fprintf(places\_file, "%s\n", places[i].name);

}

fclose(places\_file);

}

FILE \*bookings\_file = fopen("bookings.txt", "w");

if (bookings\_file) {

for (int i = 0; i < num\_bookings; i++) {

fprintf(bookings\_file, "%s %s %s %s\n", bookings[i].place\_name, bookings[i].customer\_name, bookings[i].booking\_date, bookings[i].booking\_time);

}

fclose(bookings\_file);

}

}

// Function to load places and bookings from files

void load\_data() {

FILE \*places\_file = fopen("places.txt", "r");

if (places\_file) {

while (fscanf(places\_file, "%s", places[num\_places].name) != EOF) {

num\_places++;

}

fclose(places\_file);

}

FILE \*bookings\_file = fopen("bookings.txt", "r");

if (bookings\_file) {

while (fscanf(bookings\_file, "%s %s %s %s", bookings[num\_bookings].place\_name, bookings[num\_bookings].customer\_name, bookings[num\_bookings].booking\_date, bookings[num\_bookings].booking\_time) != EOF) {

num\_bookings++;

}

fclose(bookings\_file);

}

}

// Function to add a place

void add\_place() {

if (num\_places >= MAX\_PLACES) {

printf("Cannot add more places.\n");

return;

}

printf("Enter the name of the place: ");

scanf("%s", places[num\_places].name);

num\_places++;

printf("Place added successfully.\n");

}

// Function to edit a place

void edit\_place() {

if (num\_places == 0) {

printf("No places available to edit.\n");

return;

}

printf("Enter the index of the place you want to edit: ");

int place\_index;

scanf("%d", &place\_index);

place\_index--;

if (place\_index >= 0 && place\_index < num\_places) {

printf("Enter the new name for the place: ");

scanf("%s", places[place\_index].name);

printf("Place updated successfully.\n");

}

else {

printf("Invalid place index.\n");

}

}

// Function to delete a place

void delete\_place() {

if (num\_places == 0) {

printf("No places available to delete.\n");

return;

}

printf("Enter the index of the place you want to delete: ");

int place\_index;

scanf("%d", &place\_index);

place\_index--;

if (place\_index >= 0 && place\_index < num\_places) {

// Shift places in the array to remove the deleted place

for (int i = place\_index; i < num\_places - 1; i++) {

strcpy(places[i].name, places[i + 1].name);

}

num\_places--;

printf("Place deleted successfully.\n");

} else {

printf("Invalid place index.\n");

}

}

// Function to view all places

void view\_places() {

printf("Available Places:\n");

for (int i = 0; i < num\_places; i++) {

printf("%d. %s\n", i + 1, places[i].name);

}

}

// Function to cancel a booking

void cancel\_booking() {

if (num\_bookings == 0) {

printf("No bookings to cancel.\n");

return;

}

printf("Enter your name: ");

char customer\_name[100];

scanf("%s", customer\_name);

int found = 0; // To check if a booking was found for the customer

for (int i = 0; i < num\_bookings; i++) {

if (strcmp(bookings[i].customer\_name, customer\_name) == 0) {

printf("Booking %d:\n", i + 1);

printf("Place: %s\n", bookings[i].place\_name);

printf("Date: %s\n", bookings[i].booking\_date);

printf("Time: %s\n", bookings[i].booking\_time);

printf("Do you want to cancel this booking? (1 for Yes, 0 for No): ");

int cancel\_choice;

scanf("%d", &cancel\_choice);

if (cancel\_choice == 1) {

for (int j = i; j < num\_bookings - 1; j++) {

strcpy(bookings[j].place\_name, bookings[j + 1].place\_name);

strcpy(bookings[j].customer\_name, bookings[j + 1].customer\_name);

strcpy(bookings[j].booking\_date, bookings[j + 1].booking\_date);

strcpy(bookings[j].booking\_time, bookings[j + 1].booking\_time);

}

num\_bookings--;

printf("Booking canceled successfully.\n");

} else {

printf("Booking not canceled.\n");

}

found = 1;

break; // Exit the loop after processing the booking

}

}

if (!found) {

printf("No bookings found for the customer: %s\n", customer\_name);

}

}

// Function to book a place

void book\_place() {

if (num\_places == 0) {

printf("No places available for booking.\n");

return;

}

printf("Enter your name: ");

char customer\_name[100];

scanf("%s", customer\_name);

printf("Enter the index of the place you want to book: ");

int place\_index;

scanf("%d", &place\_index);

place\_index--;

if (place\_index >= 0 && place\_index < num\_places) {

printf("Enter booking date: ");

char booking\_date[20];

scanf("%s", booking\_date);

printf("Enter booking time: ");

char booking\_time[20];

scanf("%s", booking\_time);

strcpy(bookings[num\_bookings].place\_name, places[place\_index].name);

strcpy(bookings[num\_bookings].customer\_name, customer\_name);

strcpy(bookings[num\_bookings].booking\_date, booking\_date);

strcpy(bookings[num\_bookings].booking\_time, booking\_time);

num\_bookings++;

printf("Booking successful!\n");

} else {

printf("Invalid place index.\n");

}

}

// Function to view all bookings

void view\_bookings() {

printf("Bookings: \n\n");

for (int i = 0; i < num\_bookings; i++) {

printf("Booking %d:\n", i + 1);

printf("Place: %s\n", bookings[i].place\_name);

printf("Customer: %s\n", bookings[i].customer\_name);

printf("Date: %s\n", bookings[i].booking\_date);

printf("Time: %s\n", bookings[i].booking\_time);

printf("\n");

}

}

int main() {

load\_data(); // Load data from files at the beginning

while (1) {

printf("\nWelcome to the Booking System!\n");

printf("1. Admin Login\n");

printf("2. Customer Login\n");

printf("3. Quit\n");

printf("Enter your choice: ");

int choice;

scanf("%d", &choice);

switch (choice) {

case 1:

// Admin section

if (admin\_login()) {

while (1) {

printf("\nAdmin Menu:\n");

printf("1. Add Place\n");

printf("2. Edit Place\n");

printf("3. Delete Place\n");

printf("4. View Places\n");

printf("5. View Bookings\n");

printf("6. Logout\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

add\_place();

break;

case 2:

edit\_place();

break;

case 3:

delete\_place();

break;

case 4:

view\_places();

break;

case 5:

view\_bookings();

break;

case 6:

printf("Admin logged out.\n");

break;

default:

printf("Invalid choice.\n");

}

save\_data(); // Save data to files after each operation

if (choice == 6) {

break;

}

}

} else {

printf("Invalid admin credentials.\n");

}

break;

case 2:

// Customer section

printf("Welcome, Customer!\n");

printf("1. View Places\n");

printf("2. Book Place\n");

printf("3. Cancel Booking\n");

printf("4. Quit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

view\_places();

break;

case 2:

book\_place();

break;

case 3:

cancel\_booking();

break;

case 4:

printf("Customer logged out.\n");

break;

default:

printf("Invalid choice.\n");

}

break;

case 3:

printf("Exiting the Booking System. Goodbye!\n");

save\_data(); // Save data to files before exiting

return 0;

default:

printf("Invalid choice.\n");

}

}

return 0;

}

# CONCLUSION

Both administrators and users may profit from the Travelling Assistant Booking System. It simplifies the process of making reservations for travel by offering administrators with powerful administration capabilities and a user-friendly interface for customers. For all users involved in travel planning and booking, the system offers a seamless and effective experience because to its focus on CRUD processes, thorough content, and secure authentication.